Embracing Failure: Recovery Oriented Computing (ROC)

RECOVERY-ORIENTED COMPUTING

Dave Patterson

University of California at Berkeley

patterson@cs.berkeley.edu

April 2002

roc.cs.berkeley.edu/talks

Outline

- **The past:** where we have been
- **The present:** new realities and challenges
  - Tie to supercomputing
- **A future:** Recovery-Oriented Computing (ROC)
- **ROC techniques and principles**

The past: research goals and assumptions of last 15 years

- **Goal #1: Improve performance**
- **Goal #2: Improve performance**
- **Goal #3: Improve cost-performance**
- **Assumptions**
  - Humans are perfect (they don't make mistakes during installation, wiring, upgrade, maintenance or repair)
  - Software will eventually be bug free
    (Hire better programmers!)
  - Hardware MTBF is already very large (~100 years between failures), and will continue to increase
  - Maintenance costs irrelevant vs. Purchase price
    (maintenance a function of price, so cheaper helps)

After 15 years of research on price-performance, what's next?

- **Services as model for future of IT**
- **Availability is now vital metric for services**
  - – near-100% availability is becoming mandatory
    - » for e-commerce, enterprise apps, online services, ISPs
  - – but, service outages are frequent
    - » 65% of IT managers report that their websites were unavailable to customers over a 6-month period
      - 25%: 3 or more outages
  - – outage costs are high
    - » social effects: negative press, loss of customers who "click over" to competitor

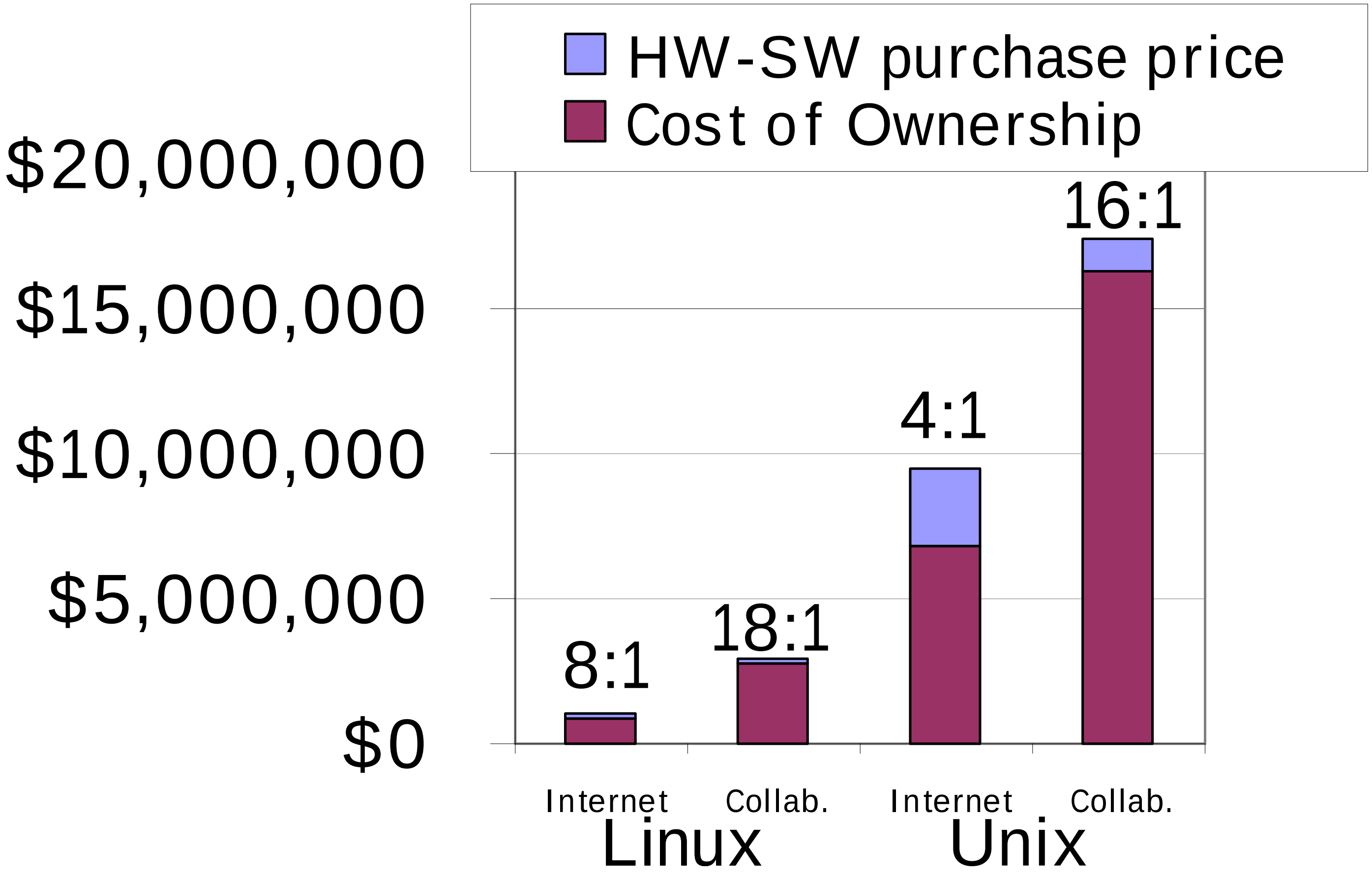
Source: InternetWeek 4/3/2000

Downtime Costs (per Hour)

Brokerage operations	$6,450,000
Credit card authorization	$2,600,000
Ebay (1 outage 22 hours)	$225,000
Amazon.com	$180,000
Package shipping services	$150,000
Home shopping channel	$113,000
Catalog sales center	$90,000
Airline reservation center	$89,000
Cellular service activation	$41,000
On-line network fees	$25,000
ATM service fees	$14,000

Sources: InternetWeek 4/3/2000 + *Fibre Channel: A Comprehensive Introduction*, R. Kembel 2000, p.8. "...based on a survey done by Contingency Planning Research."

Total Cost of Ownership: Ownership vs. Purchase



- **HW/SW decrease vs. Salary Increase**
  - 142 sites, 1200-7600 users/site, $2B/yr sales

Source: "The Role of Linux in Reducing the Cost of Enterprise Computing", IDC white paper, sponsored by Red Hat, by Al Gillen, Dan Kusnetzky, and Scott McLaron, Jan. 2002, available at www.redhat.com

What have we learned from past projects?

- **Maintenance of machines (with state) expensive**
  - ~5X to 10X cost of HW/SW
  - Stateless machines can be trivial to maintain (Hotmail)
- **System admin keeps system available; 1/3 to 1/2 of Cost of Ownership?**
  - System + clever human working during failure = uptime (failure often occurs during upgrades, reconfiguration)
  - Also growth plans, user training, fix performance bugs
- **Know how evaluate (performance and cost)**
  - Run system against workload, measure, innovate, repeat
  - Benchmarks standardize workloads, lead to competition, evaluate alternatives; turns debates into numbers
- **What are 21st century research challenges? Says who?**

IBM Research (Dean's talk Wed)

- Overview: Computing is too hard. It's time we stop our preoccupation with faster and more powerful and start making them smarter.
- The Solution: "Autonomic Computing" a systemic view of computing modeled after a self-regulating biological system; largely self-managing, self-diagnostic. User perspective:
  - Flexible The system will be able to sift data via a platform- and device-agnostic approach
  - Accessible The nature of the autonomic system is that it is always on
  - Transparent The system will perform its tasks and adapt to a user's needs without dragging the user into the intricacies of its workings

Source: www.research.ibm.com/autonomic/

A New Research Manifesto

- **Synergy with Humanity**
  - Build systems that work well with people who operate them, both end users on client computers and operators on server computers
- **Dependable Systems**
  - Build systems that world can safely depend upon
- **Secure Systems that Protect Privacy**
  - Need to help make society secure without compromising privacy of individuals
- **ROC project aimed at services at Internet sites, focus so far on operator synergy & dependability**

What does this have to do with Supercomputing?

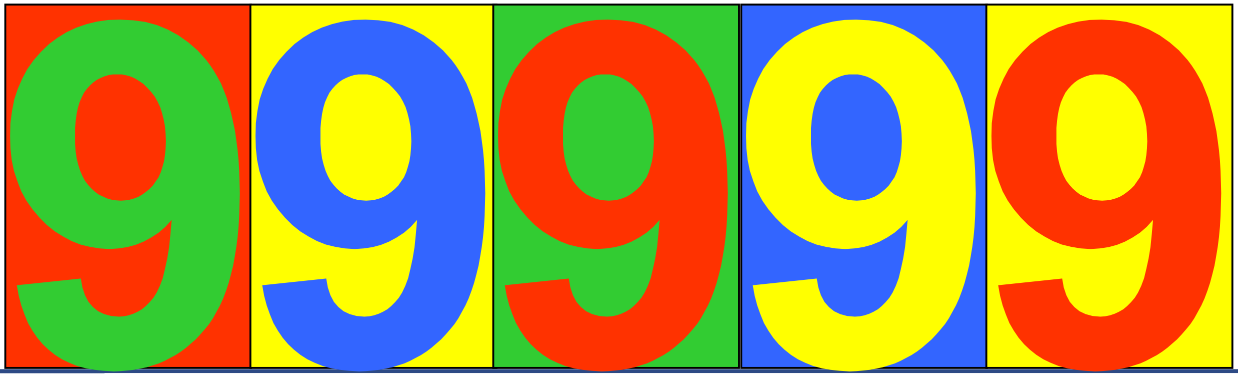
- **Supercomputing cycles are a natural service**
- **ASCI: taxpayers pay for new HW/SW based on peak performance per $100M**
  - OK to deliver single digit % of peak performance???
  - Spend programmer/scientists resources to increase HW peak performance (e.g., inadequate memory, I/O)
- **Beowulf clusters: get $ for HW/SW as part of science/engineering research grant**
  - People costs 5X-10X HW/SW costs due to lowered cost of PCs, networks, open source SW
  - Improved cost of ownership, dependability may be worthwhile investment vs. peak performance
  - Beowulf clusters better model of the future?

Where are we today? Dependability

- **Failures are common**
  - Well designed and manufactured HW: >1% fail/year
  - Well designed and tested SW: > 1 bug / 1000 lines
  - Well trained people doing difficult tasks: up to 10%
  - Well run co-location site (e.g., Exodus):
    1 power failure per year, > 1 network outage per year
  - Denial of service attacks => routine event

Dependability: Claims of 5 9s?

- 99.999% availability from telephone company?
  - **AT&T switches < 2 hours of failure in 40 years**
- Cisco, HP, Microsoft, Sun … claim 99.999% availability claims (5 minutes down / year) in marketing/advertising
  - **HP-9000 server HW and HP-UX OS can deliver 99.999% availability guarantee “in certain pre-defined, pre-tested customer environments”**
  - **Environmental? Application? Operator?**

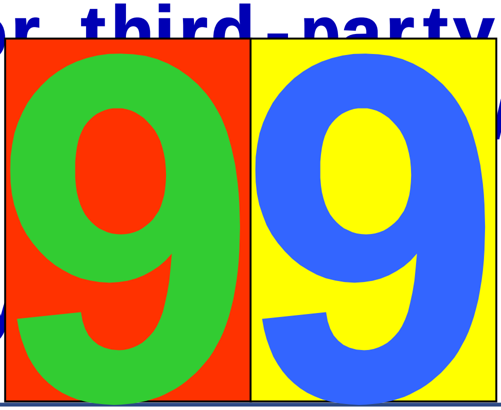


5 9s from Jim Gray’s talk:
“Dependability
in the Internet Era”

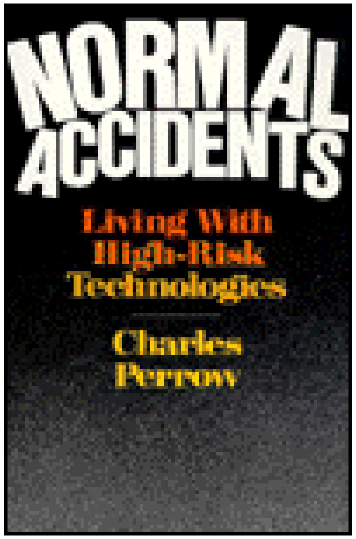
“Microsoft fingers technicians for crippling site outages”

By Robert Lemos and Melanie Austria Farmer, ZDNet News, January 25, 2001

- Microsoft blamed its own technicians for a crucial error that crippled the software giant's connection to the Internet, almost completely blocking access to its major Web sites for nearly 24 hours… a "router configuration error" had caused requests for access to the company’s Web sites to go unanswered…
- "This was an operational error and not the result of any issue with Microsoft or third-party products, nor with the securi works," a Microsoft spokesman said.
    - (5 9s possible if site stay rs!)



Learning from other fields: disasters



NORMAL ACCIDENTS

Living With High-Risk Technologies

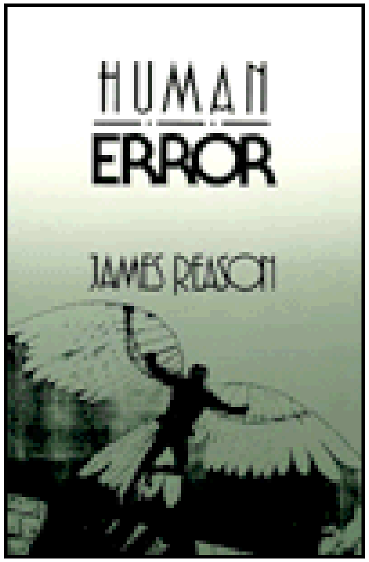
Charles Perrow

Common threads in accidents ~3 Mile Island

1. More multiple failures than you believe possible, because latent errors accumulate
2. Operators cannot fully understand system because errors in implementation, measurement system, warning systems. Also complex, hard to predict interactions
3. Tendency to blame operators afterwards (60-80%), but they must operate with missing, wrong information
4. The systems are never all working fully properly: bad warning lights, sensors out, things in repair
5. Emergency Systems are often flawed. At 3 Mile Island, 2 valves left in the wrong position; parts of a redundant system used only in an emergency. Facility running under normal operation masks errors in error handling

Charles Perrow, *Normal Accidents: Living with High Risk Technologies*, Perseus Books, 1990

Learning from other fields: human error



- **Two kinds of human error**
  - **1) slips/lapses:** errors in execution
  - **2) mistakes:** errors in planning
  - – errors can be **active** (operator error) or **latent** (design error, management error)
- **Human errors are inevitable**
  - – "humans are furious pattern-matchers"
    - » sometimes the match is wrong
  - – cognitive strain leads brain to think up least-effort solutions first, even if wrong
- **Humans can self-detect errors**
  - – about 75% of errors are immediately detected

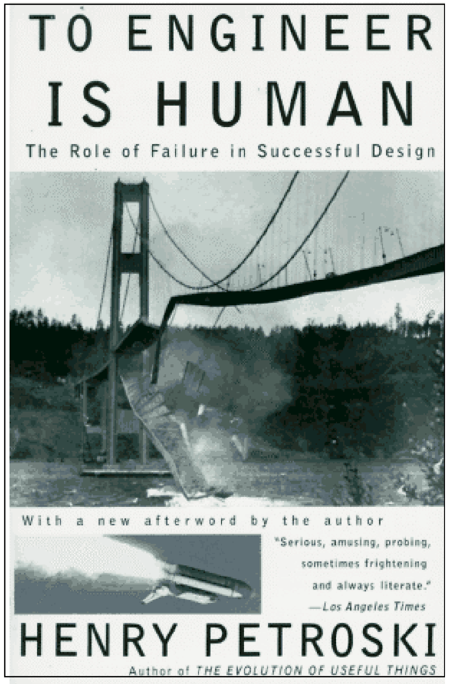
Source: J. Reason, Human Error, Cambridge, 1990.

The Automation Irony

- **Automation does not cure human error**
  - Automation shifts some errors from operator errors to design errors
    - » harder to detect/tolerate/fix design errors
  - Automation addresses the easy tasks, leaving the complex, unfamiliar tasks for the human
    - » humans are ill-suited to these tasks, especially under stress
  - Automation hinders understanding and mental modeling
    - » decreases system visibility and increases complexity
    - » operators don’t get hands-on control experience
    - » prevents building mental rules and models for troubleshooting

Learning from others: Bridges

- 1800s: 1/4 iron truss railroad bridges failed!
- Safety is now part of Civil Engineering DNA
- Techniques invented since 1800s:
  - Learn from failures vs. successes
  - Redundancy to survive some failures
  - Margin of safety 3X-6X vs. calculated load
  - (CS&E version of safety margin?)
- What will people of future think of our computers?


TO ENGINEER IS HUMAN
The Role of Failure in Successful Design
With a new afterword by the author
"Serious, amusing, probing, sometimes frightening and always literate."
—Los Angeles Times
HENRY PETROSKI
Author of THE EVOLUTION OF USEFUL THINGS

Antique Roadshow 3005 A.D.

VALTREX: Ah ha. You paid 7 million Rubex too much. My suggestion: beam it directly into the disposal cube.

These pieces of crap crashed and froze so frequently that people became violent!

Hargh!

“Worthless Piece of Crap

0 Rubex”

Summary: the present

- **We help create a brittle technology, which world depends on; will history judge IT kindly?**
- **After >20 years of working on performance, 21st Century research needs new, relevant goals**
- **A New Research Manifesto**
  - Synergy with Humanity
    - » gadgets and servers work well with people who operate them
  - Dependable Systems
    - » Build systems that world can safely depend upon
  - Secure Systems that Protect Privacy
    - » Make society secure without compromising privacy

Outline

- **The past:** where we have been
- **The present:** new realities and challenges
- **A future:** Recovery-Oriented Computing (ROC)
- **ROC techniques and principles**

Recovery-Oriented Computing Philosophy

> **"If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time"**
>
> *— Shimon Peres ("Peres's Law")*

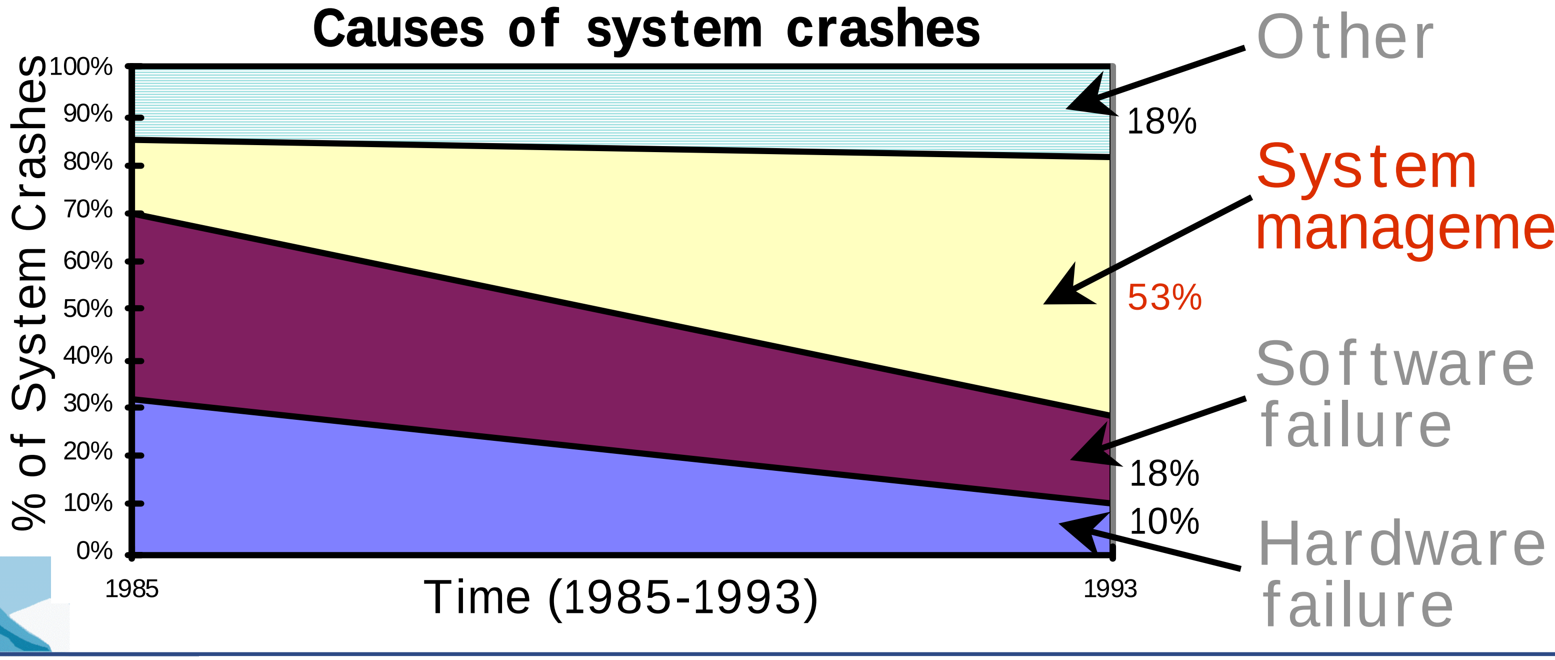
- **People/HW/SW failures are facts, not problems**
- **Recovery/repair is how we cope with them**
- **Improving recovery/repair improves availability**
  - UnAvailability = $\frac{\text{MTTR}}{\text{MTTF}}$ *(assuming MTTR much less than MTTF)*
  - 1/10th MTTR just as valuable as 10X MTBF; maybe more?
- **ROC also helps with maintenance/TCO**
  - since major Sys Admin job is recovery after failure
- **Since cost of ownership >> cost of HW/SW, spend disk/DRAM/CPU resources for recovery**

ROC approach

1. **Collect data to see why services fail**
2. **Create benchmarks to measure recovery**
   - use failure data as workload for benchmarks
   - benchmarks inspire and enable researchers / humiliate companies to spur improvements
3. **Margin of Safety in CS&E?**
4. **Create and Evaluate techniques to help recovery**
   - identify best practices of Internet and Enterprise services
   - ROC focus on fast repair (they are facts of life) vs. FT focus longer time between failures (problems)

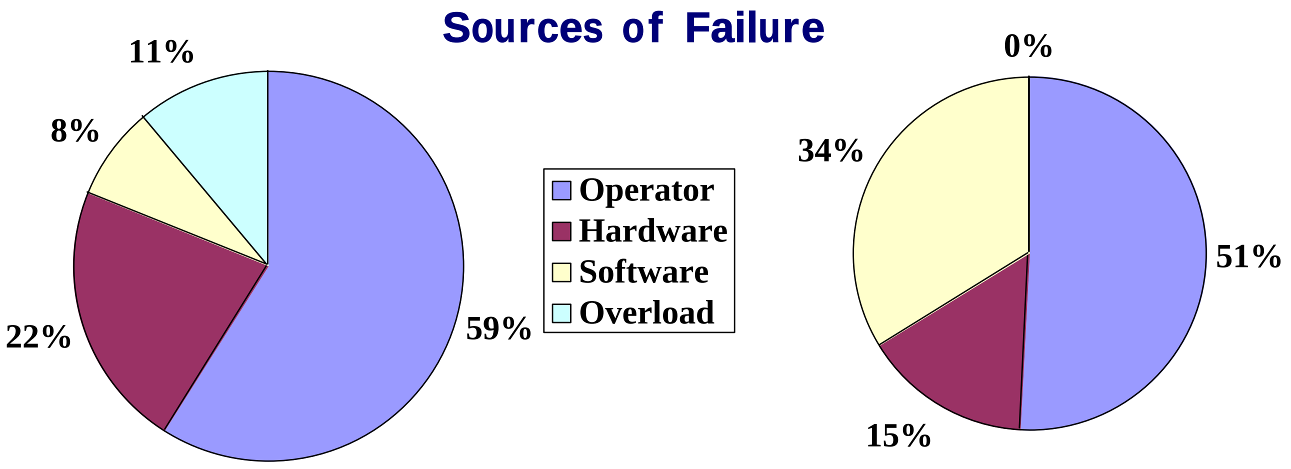
ROC Part I: Failure Data
Lessons about human operators

- **Human error is largest single failure source**
  - HP HA labs: human error is #1 cause of failures (2001)
  - Oracle: half of DB failures due to human error (1999)
  - Gray/Tandem: 42% of failures from human administrator errors (1986)
  - Murphy/Gent study of VAX systems (1993):



Human error

- **Human operator error is the leading cause of dependability problems in many domains**



Public Switched Telephone Network

Average of 3 Internet Sites

- **Operator error cannot be eliminated**
  - humans inevitably make mistakes: “to err is human”
  - *automation irony* tells us we can’t eliminate the human

Source: D. Patterson et al. Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies, UC Berkeley Technical Report UCB//CSD-02-1175, March 2002.

ROC Part 1: Failures Data Collection (so far)

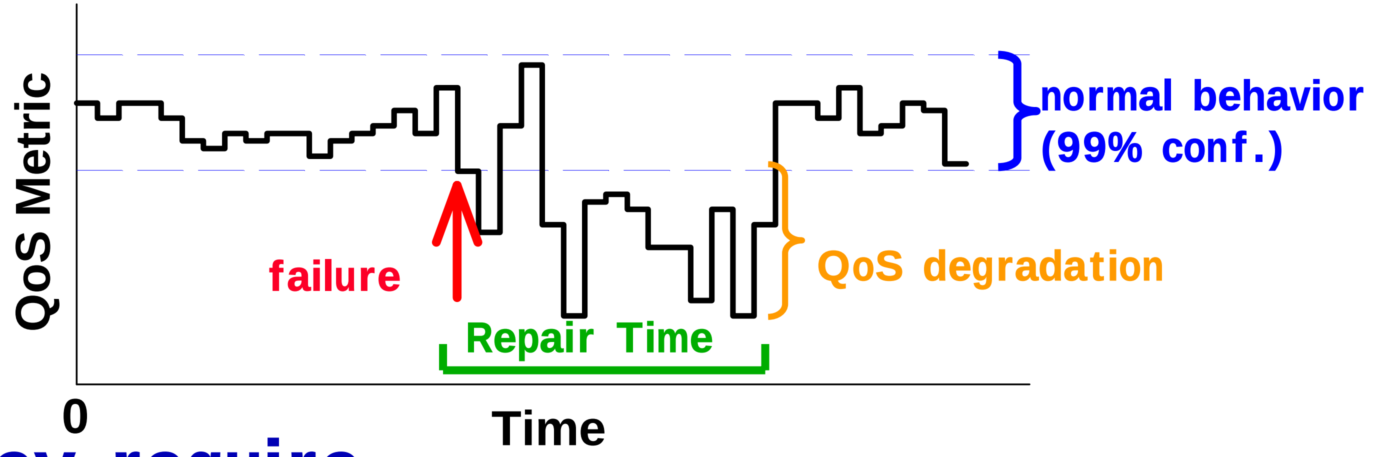
- **Humans substantial cause of failures**
  - As end users
  - As operators
- **Internet sites also challenged by network outages**
  - Significant outages due to relying on collocation site facilities
  - Problem diagnosis/repair difficult when components maintained by independent entities
- **Very interested in getting more data (under NDA if desired) if you know where to get it**

ROC Part 2: Recovery benchmarks

- **Traditional benchmarks focus on performance**
  - ignore ACME goals
  - assume perfect hardware, software, human operators
- **20th Century Winner: Peak Performance?**
- **21st Century Winner: Delivered Performance in presence of failures, variable behavior?**
- **New benchmarks needed to drive progress toward Dependability, Robustness**
  - How else convince developers, customers to adopt new technology?
  - How else enable researchers to find new challenges?

Recovery benchmarking 101

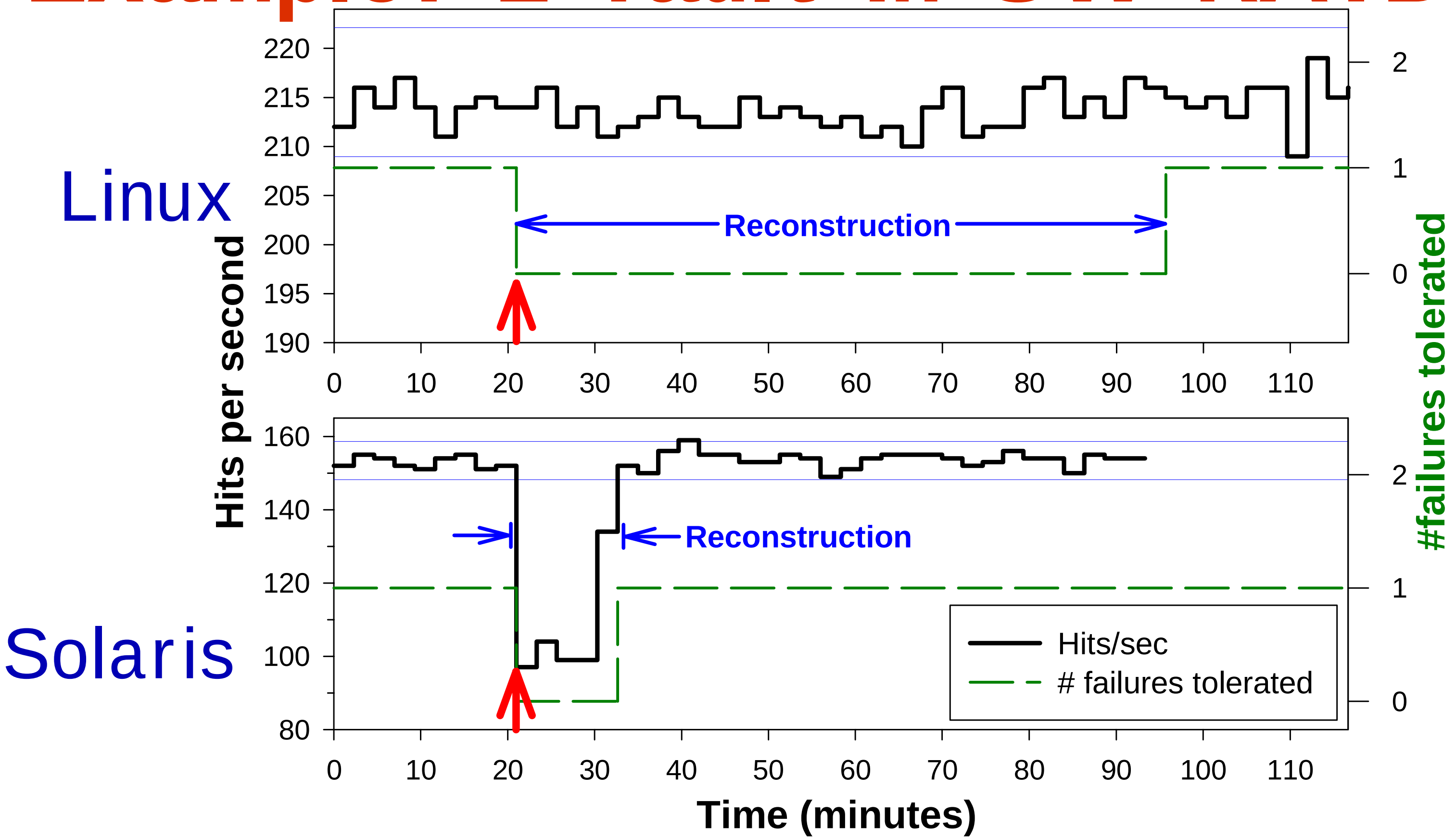
- **Recovery benchmarks quantify system behavior under failures, maintenance, recovery**



- **They require**
  - A realistic workload for the system
  - Quality of service metrics and tools to measure them
  - Fault-injection to simulate failures
  - Human operators to perform repairs

Source: A. Brown, and D. Patterson, "Towards availability benchmarks: a case study of software RAID systems," *Proc. USENIX*, 18-23 June 2000

Example: 1 fault in SW RAID



- Compares Linux and Solaris reconstruction
  - **Linux:** minimal performance impact but longer window of vulnerability to second fault
  - **Solaris:** large perf. impact but restores redundancy fast
  - **Windows**: does not auto-reconstruct!

Margin of Safety in CS&E?

- **Like Civil Engineering, never make dependable systems until add margin of safety (“margin of ignorance”) for what we don’t (can’t) know?**
  - – Before: design to tolerate expected (HW) faults
- **Marketing claims available 5 9s (99.999%) but customers achieve 2-3 9s (99% to 99.9%)**
- **Perhaps we need to “over engineer” by a 1-2 9’s to deliver what we claim?**
- **E.g., RAID 5 v. RAID 6 (double failure OK)**
  - – Temperature, vibration causing failure before repair
  - – Plus operator removing good disk vs. bad disk
- **Extra resources to mask errors + to “time travel” before SW or human fault?**

ROC Part 4: Create/Evaluate ROC Techniques

- **Need a theory on constructing dependable, maintainable sites for networked services**
  - Document best practices of successful sites?
- **Need a theory on good design for operators as well as good design for end users**
  - Airplane Analogy: user interface to passengers (747) vs. user interface to pilots (Cessna)
- **Need better definition of failure**
  - Need IT equivalent of PSTN "blocked calls"?
    - » PSTN switches required to collect blocked calls; why don't Internet switches collect failures?
  - Failure > unavailable for 100% of users: (e.g., available to 10% of users is not "up")

Safe, forgiving for operator?

- **Expect human error and tolerate it**
  - protect system data from human error
  - allow mistakes to be easily reversed
- **Allow human operator to learn naturally**
  - "mistakes are OK": design to encourage exploration, experimentation
- **Make training on real system an everyday process**
- **Match interfaces to human capabilities**
- **Automate tedious or difficult tasks, but retain manual procedures**
  - Encourage periodic use of manual procedures to increase familiarity

Partitioning and Redundancy?

- **System is Partitionable**
  - To isolate faults
  - To enable online repair/recovery
  - To enable online HW growth/SW upgrade
  - To enable operator training/expand experience on portions of real system without fear of system failure
  - Techniques: Geographically replicated sites, Virtual Machine Monitors
- **System is Redundant**
  - Sufficient HW redundancy/Data replication => part of system down but satisfactory service still available
  - Enough to survive 2nd (nth?) failure during recovery
  - Techniques: RAID-6, N-copies of data

Input Insertion for Detection?

- **System enables input insertion, output check of all modules (including fault insertion)**
  - To check module sanity to find failures faster
  - To test correctness of recovery mechanisms
    - » insert (random) faults and known-incorrect inputs
    - » also enables availability benchmarks
  - To expose and remove latent errors from system
  - To train/expand experience of operator
    - » Periodic reports to management on skills
  - To discover if warning systems are broken
    - » How else tell?
  - To simplify use of Recovery benchmarks
- **Example: FIG - Fault Insertion in Glibc**
  - <10% overhead finds strange behavior even in mature software when invoke errors

Aid Diagnosis?

- **System assists human in diagnosing problems**
  - Root-cause analysis to suggest possible failure points
    - » Track resource dependencies of all requests
    - » Correlate symptomatic requests with component dependency model to isolate culprit components
  - "health" reporting to detect failed/failing components
    - » Failure information, self-test results propagated upwards
  - Don't rely on things connected according to plans
    - » Example: Discovery of network, power topology
- **Example: Pinpoint – modify J2EE to trace modules used and record success/fail of trace, then use standard data mining to discover failed module; 8% overhead, don't need architectural model, yet very accurate**

Refresh via Recovery?

- **Many Internet services refresh system by periodic restart**
- **"Recursive Recovery" (Candea, Fox) restarts optimal number of components of system**
- **Look at dependence chain during recovery to see if can reorganize to reduce recovery time**
- **Example: Mercury satellite ground station**
  - Average 5X reduction in recovery time
  - Nonlinear return: fast recovery implies don't lose track of satellite during pass vs. greater MTTF

Source: G. Candea and A. Fox, "Recursive Restartability: Turing the Reboot Sledgehammer into a scalpel," ***8th Workshop on Hot Topics in Operating Systmes (HotOS-VIII)*****, May 2001**

Support Operator Trial & Error?

- **Time travel for system operators**
- **Three R’s for recovery**
  - **Rewind:** roll all system state backwards in time
  - **Repair:** change system to prevent failure
    - » e.g., fix latent error, retry unsuccessful operation, install preventative patch
  - **Replay:** roll system state forward, replaying end-user interactions lost during rewind
- **All three R’s are critical**
  - rewind enables undo
  - repair lets user/administrator fix problems
  - replay preserves updates, propagates fixes forward

Example 3R's scenarios

- **Direct operator errors**
  - system misconfiguration
    - configuration file change, email filter installation, ...
  - accidental deletion of data
    - "rm –rf /", deleting a user's email spool, reversed copy during data reorganization, ...
- **Retroactive repair**
  - mitigate external attacks
    - retroactively install virus/spam filter on email server; effects are squashed on replay
  - repair broken software installations
    - mis-installed software patch, installation of software that corrupts data, software upgrade that slows performance
- **Undo spends excess disk capacity to offer safety margin via time travel**

ROC Status

- Papers that layout philosophy and initial results for recovery benchmarks, failure data collection, FIG library insertion, Pinpoint diagnosis, Mercury recursive recovery
- Building Email prototype for operator undo
- Plan on Email system this year using all ROC techniques, then benchmark recovery vs. commercial systems
- Need 2nd application to make ROC more convincing? suggestions welcome
  - .Net or J2EE Middleware? Involves Storage? Geographically Distributed?

ROC Summary

- **21st Century Research challenge is Synergy with Humanity, Dependability, Security/Privacy**
- **2002: Peres's Law greater than Moore's Law?**
  - Must cope with fact that people, SW, HW fail
  - Industry: may soon compete on recovery time v. peak
- **Recovery Oriented Computing is one path for operator synergy, dependability for servers**
  - Failure data collection + Benchmarks to evaluate
  - Partitioning, Redundancy, Diagnosis, Partial Recovery, Input/Fault Insertion, Undo, Margin of Safety (spare 9s)
- **Significantly reducing MTTR (people/SW/HW) => better Dependability & Cost of Ownership**

Interested in ROCing?

- **More research opportunities than 2 university projects can cover. Many could help with:**
  - Failure data collection, analysis, and publication
  - Create/Run Availability, Maintainability benchmarks: compare (by vendor) databases, files systems, routers, …
  - Invent, evaluate techniques to reduce MTTR and TCO in computation, storage, and network systems
  - (Lots of low hanging fruit)

“If it’s important,
how can you say it’s impossible if you don’t try?”

Jean Monnet, a founder of European Union

http://ROC.cs.berkeley.edu

Can DOE ever kick the desktop microprocessor habit?

- Not if use peak performance to make decision
- Amdahl's Law says what happens if improve one piece of system without improving others
- Embedded computing technology offers fast turnaround, customized silicon
  - Assemble predesigned components + special sauce
- For example, 6 grad students designed 1 GFLOPS per watt vector microprocessor with 13MB low latency (8 clocks) memory on-chip in 2.5 years for $5M (VIRAM-1)
  - CPU from MIPS, DRAM from IBM, FPU from MIT, ...

BACKUP SLIDES

Failure Data: Public Switched Telephone Network (PSTN)

- Detailed telephone service failure data available from the Federal Communications Commission (FCC)
  - Required by law for outages affecting 30,000 people or lasting at least 30 minutes
  - 3 ways to report

1. Outage and reason (direct vs. root cause)
   - But how big an outage?
2. Length of outage * potential customers affected
   - But what if 2 AM vs. 2 PM?
3. Blocked calls: actual calls tried but unsuccessful due to outage (!)

ROC Part 2:
ACME Benchmarks (so far)

- Race to recover vs. race to finish line
- Many opportunities to compare commercial products and claims, measure value of research ideas, … with availability benchmarks
- Maintainability benchmarks involve people, but so do most research by social scientists
- Partial failures: Evaluate "Service level" benchmarks that insert faults that do not bring down entire service for all users?
- Even initial Availability benchmarks find peculiarities of systems measured
- Lots of low hanging fruit (~ early RAID days)

Availability Benchmarking Environment

- **Fault workload**
  - Must accurately reflect failure modes of real-world Internet service environments
    - » plus random tests to increase coverage, simulate Heisenbugs
  - But, no existing public failure dataset
    - » we have to collect this data
    - » a challenge due to proprietary nature of data
  - major contribution will be to collect, anonymize, and publish a modern set of failure data
- **Fault injection harness**
  - build into system: needed anyway for online verification

Software RAID: QoS behavior

- **Response to double-fault scenario**
  - a double fault results in unrecoverable loss of data on the RAID volume
  - **Linux:** blocked access to volume
  - **Windows:** blocked access to volume
  - **Solaris:** silently continued using volume, delivering *fabricated* data to application!
    - » clear violation of RAID availability semantics
    - » resulted in corrupted file system and garbage data at the application level
    - » this *undocumented* policy has serious availability implications for applications

Failure Data: 2 Internet Sites

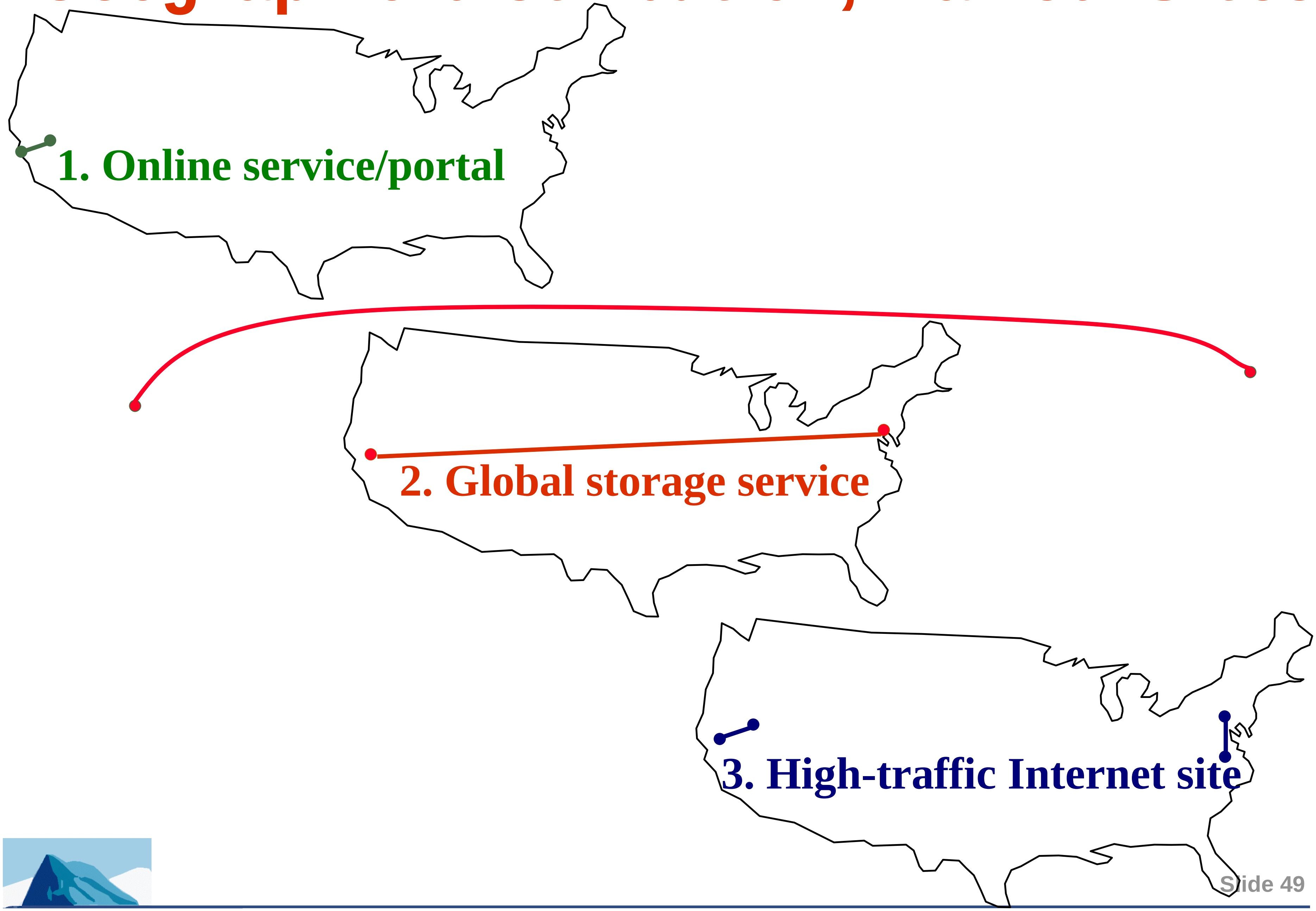
- **Global storage service**
  - ~500 machines, 4 colo. facilities + customer sites
  - all service software custom-written (x86/free OS)
  - Read/Write, more complex workload
- **High-traffic Internet site**
  - ~5000 of machines, 4 collocation facilities
  - ~100 million hits/day
  - all service software custom-written (x86/free OS)
  - Read mostly
  - Read, More HW, Software more mature
- **Looked at trouble tickets over 3-6 months**

Source: David Oppenheimer, U.C. Berkeley, in progress.

Total Cost Own. Hypothesis

- **"Moore's Law" + hypercompetitve marketplace improves cost and speed of CPUs, cost and capacity of memory and disks**
- **Morris (IBM) $3M comparison 1984 v. 2001:**
  - CPU: Minicomputer to PC, 3000X faster
  - DRAM: Memory boards to DIMMs, 3000X bigger
  - Disks: 8-inch drives to 3.5-inch drives, 4000X bigger
- **Unless avg. user demands grow with Moore's Law, a service increases in number of users**
- **HW/SW costs shrink; salaries go up over time**
- **Hypothesis: Cost of Ownership is more a function of number of users versus HW/SW $, so T.C.O. today is mostly people costs**

Geographic distribution, Paired Sites



1. Online service/portal

2. Global storage service

3. High-traffic Internet site

Automation vs. Aid?

- **Two approaches to helping**

1) Automate the entire process as a unit

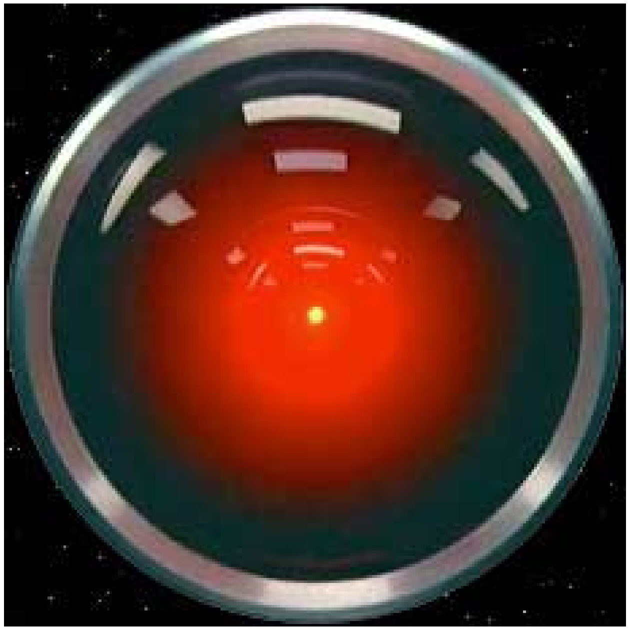
- the goal of most research into “self-healing”, “self-maintaining”, “self-tuning”, or more recently “introspective” or “autonomic” systems
- What about Automation Irony?

2) ROC approach: provide tools to let human SysAdmins perform job more effectively

- If desired, add automation as a layer on top of the tools
- What about number of SysAdmins as number of computers continue to increase?

A science fiction analogy: Autonomic vs. ROC

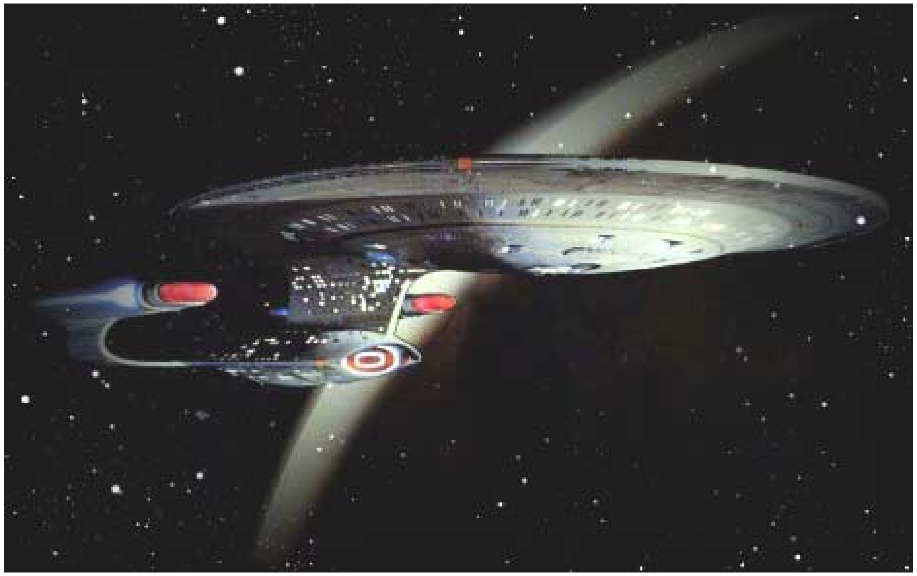
- **Autonomic approach**



HAL 9000 (2001)

- **Suffers from effects of the Automation Irony**
  - system is opaque to humans
  - only solution to unanticipated failure is to pull the plug?

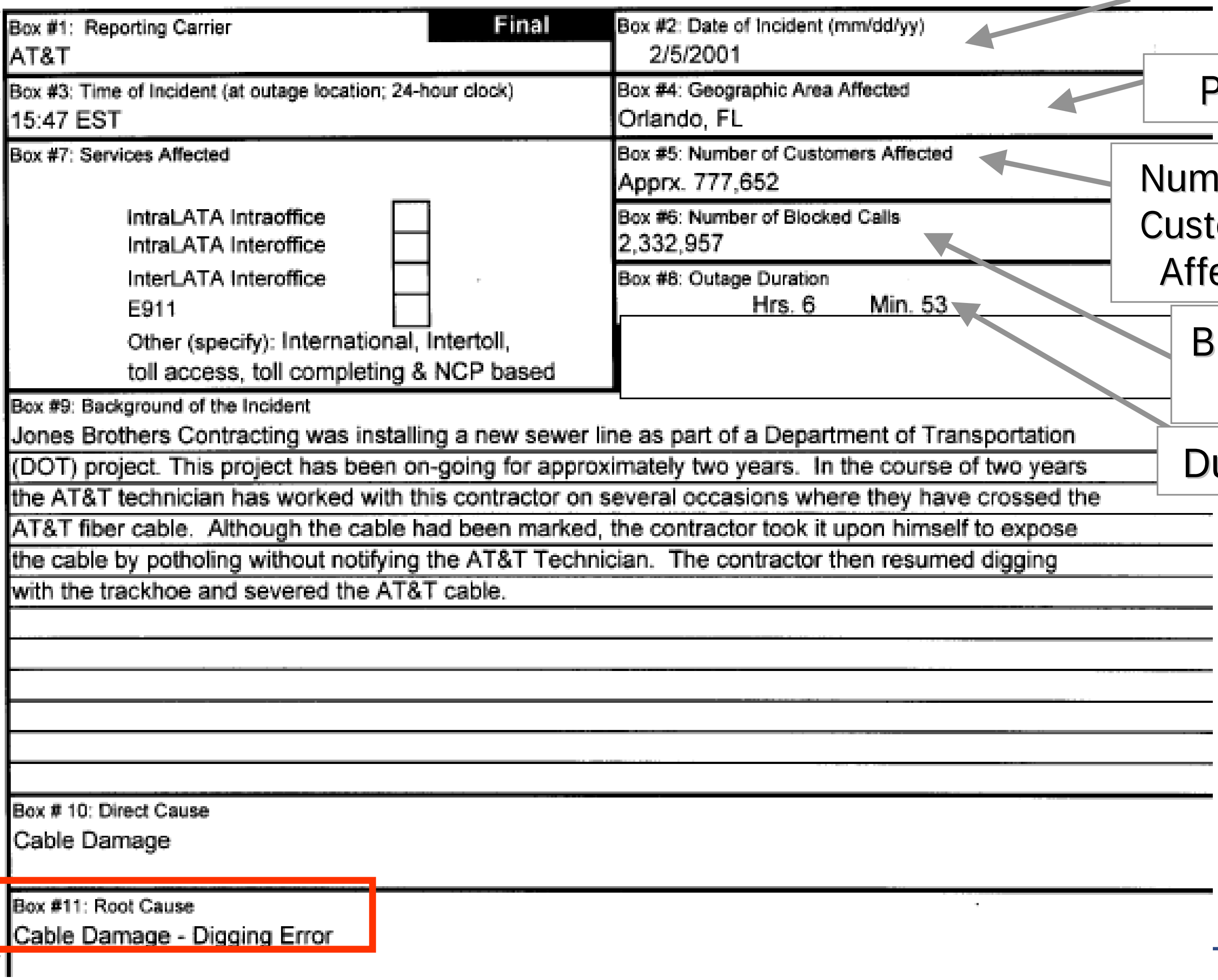
- **ROC approach**



Enterprise computer (2365)

- **24th-century engineer is like today's SysAdmin**
  - a *human* diagnoses & repairs computer problems
  - aided by diagnostic tools and understanding of system

Outage Report



WIRE LINE OUTAGE REPORTING TEMPLATE

Box #1: Reporting Carrier **Final**
AT&T

Box #2: Date of Incident (mm/dd/yy)
2/5/2001

Box #3: Time of Incident (at outage location; 24-hour clock)
15:47 EST

Box #4: Geographic Area Affected
Orlando, FL

Box #7: Services Affected

IntraLATA Intraoffice □
IntraLATA Interoffice □
InterLATA Interoffice □
E911 □
Other (specify): International, Intertoll,
toll access, toll completing & NCP based

Box #5: Number of Customers Affected
Apprx. 777,652

Box #6: Number of Blocked Calls
2,332,957

Box #8: Outage Duration
Hrs. 6 Min. 53

Box #9: Background of the Incident
Jones Brothers Contracting was installing a new sewer line as part of a Department of Transportation (DOT) project. This project has been on-going for approximately two years. In the course of two years the AT&T technician has worked with this contractor on several occasions where they have crossed the AT&T fiber cable. Although the cable had been marked, the contractor took it upon himself to expose the cable by potholing without notifying the AT&T Technician. The contractor then resumed digging with the trackhoe and severed the AT&T cable.

Box # 10: Direct Cause
Cable Damage

Box #11: Root Cause
Cable Damage - Digging Error

Company

Date

Place

Time

Number of Customers Affected

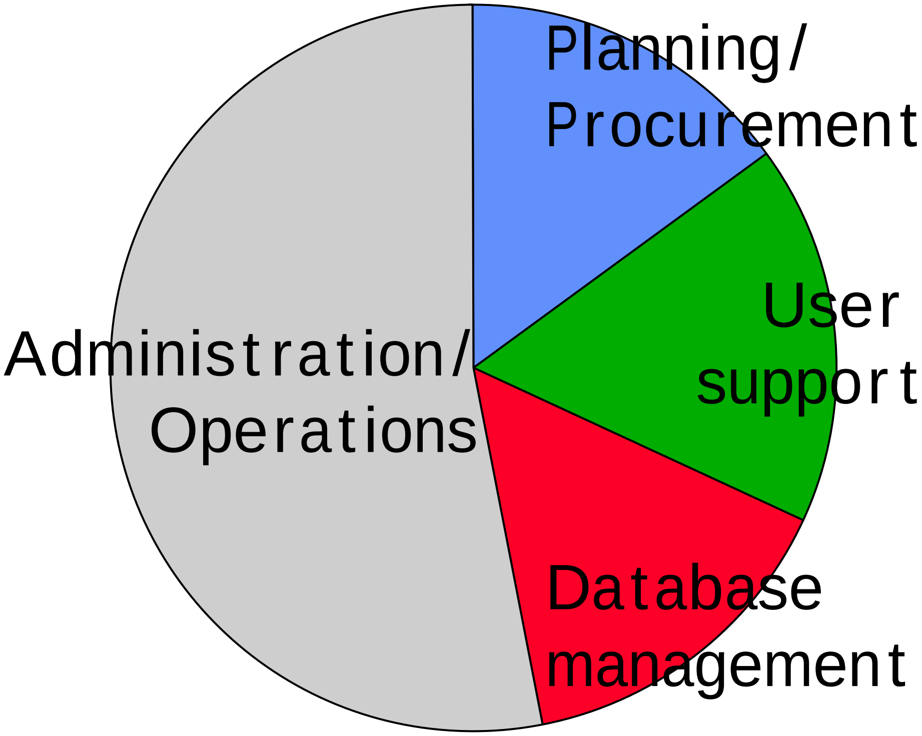
Blocked Calls

Duration

Explanation

Cause

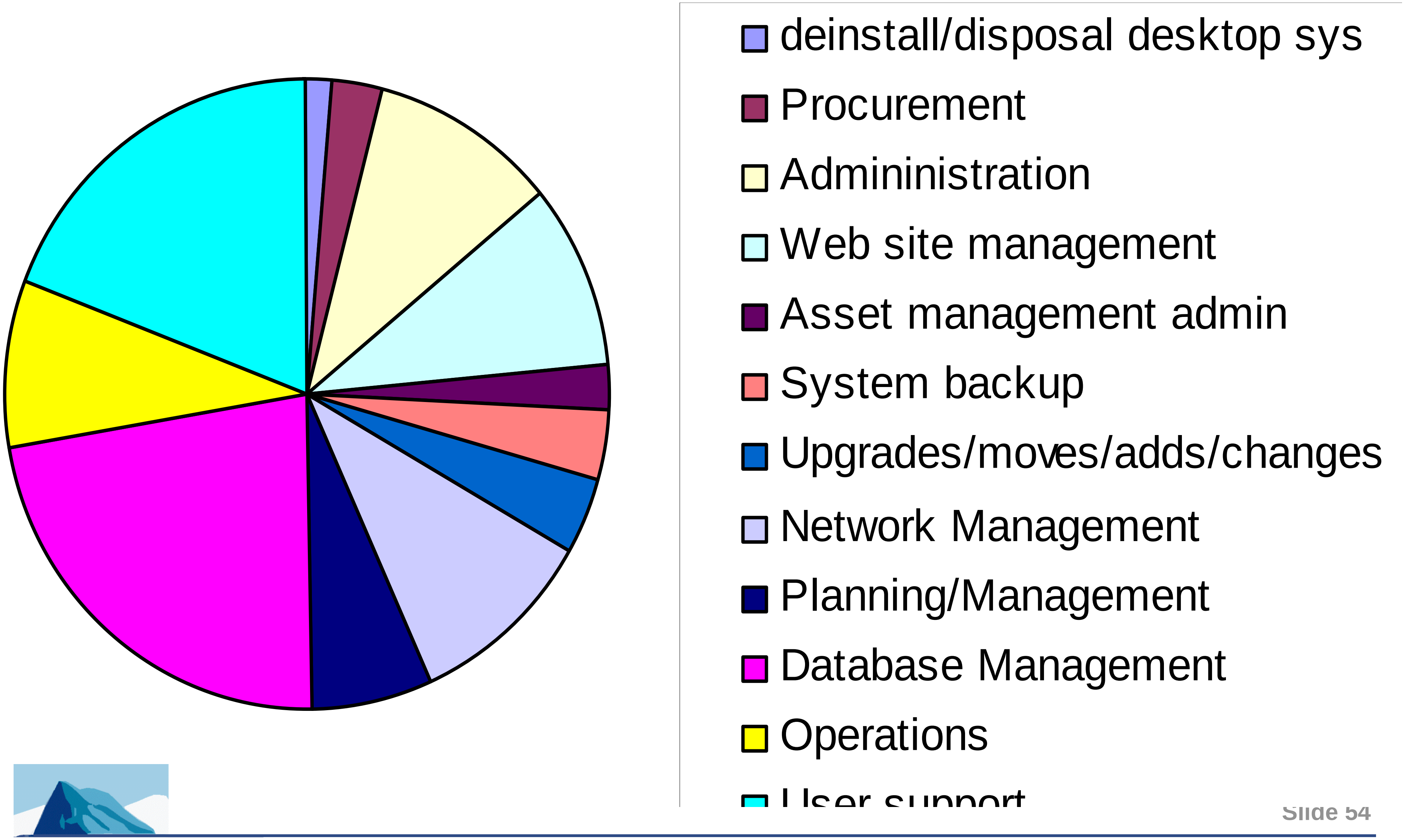
TCO breakdown (average)



Source: "The Role of Linux in Reducing the Cost of Enterprise Computing“, IDC white paper, sponsored by Red Hat, by Al Gillen, Dan Kusnetzky, and Scott McLaron, Jan. 2002, available at www.redhat.com

- **Administration/Operations**
  - Adding/deleing users
  - Tracking equipment
  - Network, Server management
  - Backup
  - Upgrades, Web site
- **Planning/Procurement**
  - Planning for upgrades
  - Buying new, disposing old
- **User support**
  - Help desk
  - Desktop troubleshooting
- **Database management**
  - Creating, adjusting, allocating DB resources

Internet x86/Linux Breakdown



Evaluating ROC: human aspects

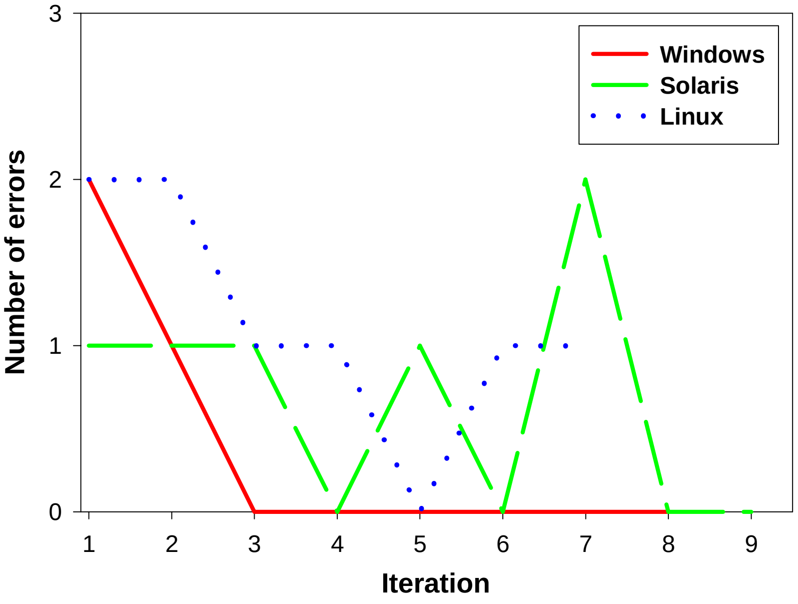
- **Must include humans in availability benchmarks**
  - to verify effectiveness of undo, training, diagnostics
  - humans act as system administrators
- **Subjects should be admin-savvy**
  - system administrators
  - CS graduate students
- **Challenge will be compressing timescale**
  - i.e., for evaluating training
- **We have some experience with these trials**
  - earlier work in maintainability benchmarks used 5-person pilot study

Example results: software RAID (2)

- **Human error rates during repair**
  - 5 trained subjects repeatedly repairing disk failures

Error type	Windows	Solaris	Linux
Fatal Data Loss	💣		💣💣
Unsuccessful Repair			💣
System ignored fatal input			💣
User Error – Intervention Required	💣	💣💣	💣
User Error – User Recovered	💣	💣💣💣💣	💣💣
Total number of trials	**35**	**33**	**31**

  - errors rates do not decline with experience
    - » early: mistakes; later: slips & lapses
    - » UI has big impact on slips & lapses



Lessons Learned from Other Cultures

- Code of Hammurabi, 1795-1750 BC, Babylon
  - 282 Laws on 8-foot stone monolith

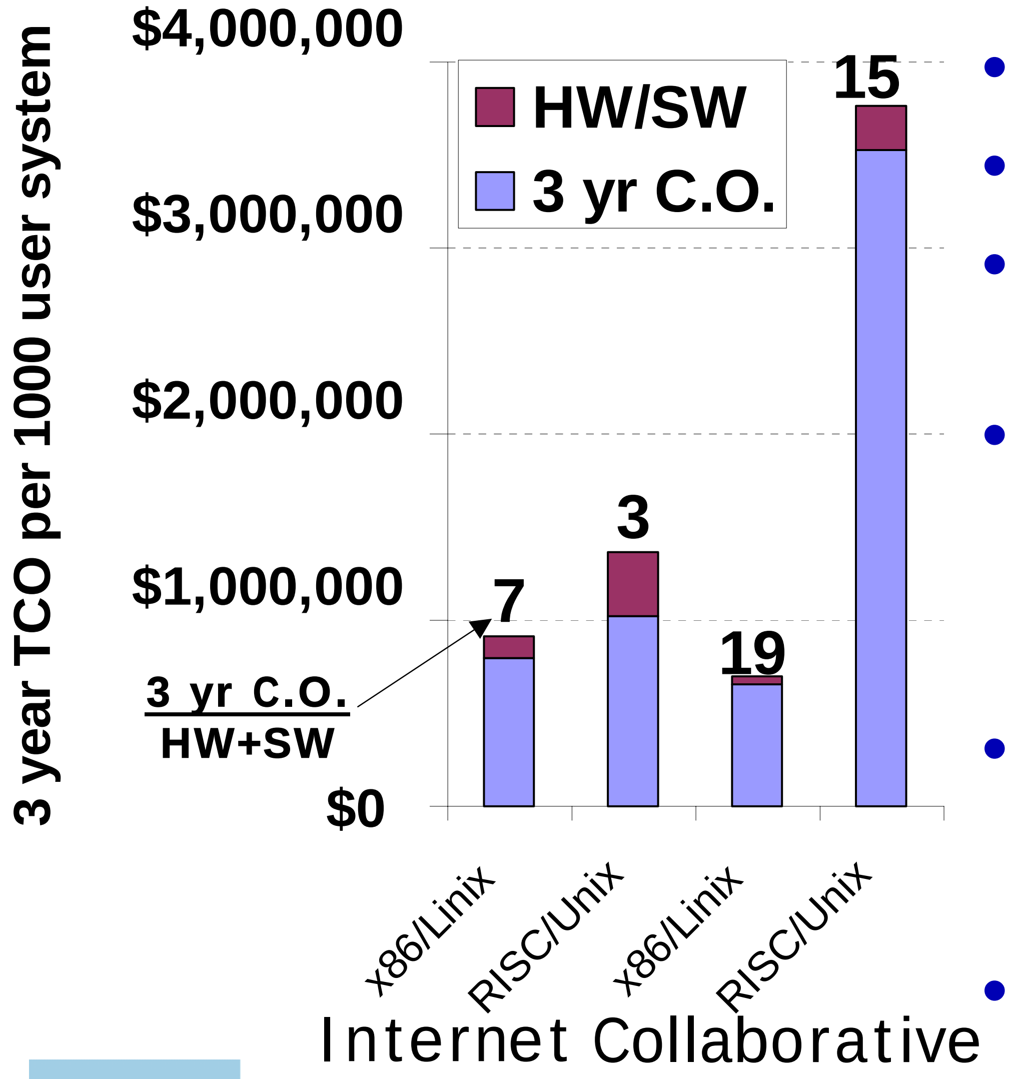
229. If a builder build a house for some one, and does not construct it properly, and the house which he built fall in and kill its owner, then that builder shall be put to death.

230. If it kill the son of the owner the son of that builder shall be put to death.

232. If it ruin goods, he shall make compensation for all that has been ruined, and inasmuch as he did not construct properly this house which he built and it fell, he shall re-erect the house from his own means.

- Do we need Babylonian quality standards?

Cost of ownership after 15 yrs of improving price-performance?



- 142 Interviews, 2H01
- $2.4B/yr avg. sales
- Avg. 3 - 12 servers, 1100 - 7600 users/site
- not included: space, power, media, comm., HW/SW support contracts, downtime
- Internet/Intranet: firewall,Web serving, Web caching, B2B, B2C
- Collaborative: calendar, email, file/database,

Source: "The Role of Linux in Reducing the Cost of Enterprise Computing“, IDC white paper, sponsored by Red Hat, by Al Gillen, Dan Kusnetzky, and Scott McLaron, Jan. 2002, available at www.redhat.com